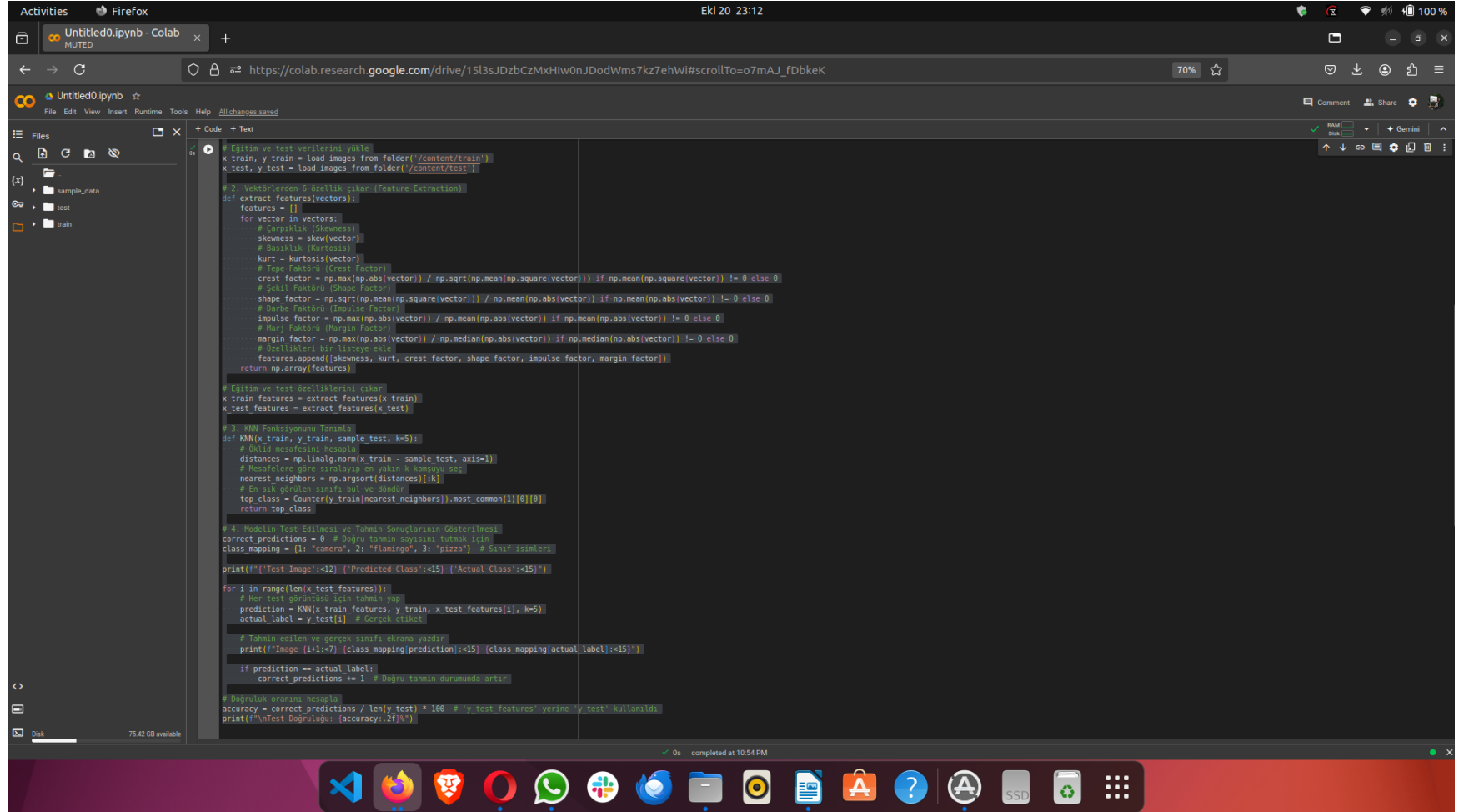


ADEM MAVANACI

152120191040

KNN SINIFLANDIRMA (HW2) Raporu

TASK1: Görüntü İşleme ve KNN Sınıflandırma



```
# Eğitim ve test verilerini yükle
x_train, y_train = load_images_from_folder('/content/train')
x_test, y_test = load_images_from_folder('/content/test')

# 2. Vektörlerden 6 özellik çıkar (Feature Extraction)
def extract_features(vectors):
    features = []
    for vector in vectors:
        # 1. Skewness (Çarpıklık)
        skewness = skew(vector)
        # 2. Kurtosis (Kurtosis)
        kurtosis = kurtosis(vector)
        # 3. Crest Factor (Crest Factor)
        crest_factor = np.max(np.abs(vector)) / np.sqrt(np.mean(np.square(vector))) if np.mean(np.square(vector)) != 0 else 0
        # 4. Shape Factor (Shape Factor)
        shape_factor = np.sqrt(np.mean(np.square(vector))) / np.mean(np.abs(vector)) if np.mean(np.abs(vector)) != 0 else 0
        # 5. Impulse Factor (Impulse Factor)
        impulse_factor = np.max(np.abs(vector)) / np.mean(np.abs(vector)) if np.mean(np.abs(vector)) != 0 else 0
        # 6. Margin Factor (Margin Factor)
        margin_factor = np.max(np.abs(vector)) / np.median(np.abs(vector)) if np.median(np.abs(vector)) != 0 else 0
        # Özellikleri bir listeye ekle
        features.append([skewness, kurtosis, crest_factor, shape_factor, impulse_factor, margin_factor])
    return np.array(features)

# Eğitim ve test özelliklerini çıkar
x_train_features = extract_features(x_train)
x_test_features = extract_features(x_test)

# 3. KNN Fonksiyonunu Tanımla
def KNN(x_train, y_train, sample_test, k=5):
    # Öklid mesafesini hesapla
    distances = np.linalg.norm(x_train - sample_test, axis=1)
    # Mesafelere göre sıralayıp en yakın k komşuyu seç
    nearest_neighbors = np.argsort(distances)[:k]
    # En sık görülen sınıfı bul ve döndür
    top_class = Counter(y_train[nearest_neighbors]).most_common(1)[0][0]
    return top_class

# 4. Modelin Test Edilmesi ve Tahmin Sonuçlarının Gösterilmesi
correct_predictions = 0 # Doğru tahmin sayısını tutmak için
class_mapping = {1: "camera", 2: "flamingo", 3: "pizza"} # Sınıf isimleri

print(f"Test Image: {x_test[0]} (Predicted Class: {y_test[0]} (Actual Class: {y_test[0]}")

for i in range(len(x_test_features)):
    # Her test görüntüsü için tahmin yap
    prediction = KNN(x_train_features, y_train, x_test_features[i], k=5)
    actual_label = y_test[i] # Gerçek etiket

    # Tahmin edilen ve gerçek sınıfı ekrana yazdır
    print(f"Image {i+1}<7> (class_mapping[prediction]: {class_mapping[prediction]} (class_mapping[actual_label]: {class_mapping[actual_label]}")

    if prediction == actual_label:
        correct_predictions += 1 # Doğru tahmin durumunda artır

# Doğruluk oranını hesapla
accuracy = correct_predictions / len(y_test) * 100 # 'y_test_features' yerine 'y_test' kullanıldı
print(f"Test Doğruluğu: (accuracy: {accuracy})")
```

Yapılan İşlemler

Görüntü işleme ve makine öğrenimi kullanarak, üç farklı sınıfa ait görüntüleri (kamera, flamingo, pizza) tanımlamak için bir model oluşturdum. İşlem aşamaları şunlardır:

- **Görüntüleri Yükleme:** Orijinal görüntü dosyalarını yüklüyorum ve her görüntüyü gri tonlamaya çevirip 128x128 boyutuna küçültüyorum.
- **Özellik Çıkarma:** Her görüntüden çarpıklık, basıklık, tepe faktörü, şekil faktörü, darbe faktörü ve marj faktörü gibi altı özellik çıkarıyorum.
- **KNN Modeli Oluşturma:** K-Nearest Neighbors (KNN) algoritması kullanarak sınıflandırma yapıyorum.
- **Model Testi:** Test setindeki görüntüler üzerinde tahminler yapıyor ve modelin doğruluğunu hesaplıyorum.

Kod Açıklaması

1. **Gerekli Kütüphanelerin İçe Aktarılması:** Görüntü işleme ve hesaplamalar için gereken kütüphaneler (os, cv2, numpy, scipy, collections) projeye dahil ediliyor.
2. **Görüntüleri Yükleme ve Küçültme:**

`def load_images_from_folder(folder_path):` Bu fonksiyon, belirtilen klasörden görüntüleri okur, gri tonlamaya çevirir ve 128x128 boyutuna küçültür. Sobel filtresi uygulanarak görüntünün kenarları belirginleştirilir. Görüntü, 1x16384 boyutunda bir vektöre dönüştürülerek listeye eklenir.

3. **Özellik Çıkarma:** Bu fonksiyon, her görüntü vektöründen belirtilen altı özellik çıkarır. Özellikler, görüntünün yapısını ve belirginliğini anlamak için kullanılır.
4. **KNN Fonksiyonu:** KNN algoritması, test görüntüsü için en yakın k komşuyu bulur ve en sık görülen sınıfı döndürür. Öklid mesafesi kullanarak komşular arasındaki uzaklık hesaplanır.
5. **Modelin Test Edilmesi:** Test verisi üzerinde döngü kurarak her görüntü için tahmin yapar ve tahmin edilen sınıf ile gerçek sınıfı karşılaştırır. Doğru tahmin sayısı hesaplanarak modelin doğruluğu elde edilir.

Karşılaşılan Sorunlar ve Çözüm Yöntemleri

1.Görüntü Yükleme Hatası: İlk denemede bazı görüntülerin yüklenememesiyle karşılaştım. Bu, belirtilen klasör yolunun hatalı olmasından kaynaklanıyordu. Çözüm olarak, klasör yolunun doğruluğunu kontrol edip, dosya okuma hatalarını göz önünde bulundurarak hata yönetimi ekledim.

2. Özellik Çıkarma Problemi: Özellik çıkarımında, görüntülerin büyüklüğünden dolayı bazı hesaplamaların sıfırdan bölünmesi hatasına yol açtı. Özellik çıkarım fonksiyonunda, ortalama değer sıfırsa bir kontrol ekleyerek bu hatayı önledim.

3. Doğruluk Oranı Düşüklüğü: Başlangıçta modelin doğruluk oranı %57 civarındaydı. Modelin doğruluğunu artırmak için daha fazla veri seti kullanmayı, görüntü augmentasyonu uygulamayı ve farklı k değerleri denemeyi planlıyorum. KNN'de kullanılan k değerinin optimal ayarı, modelin performansını artıracaktır.

Sonuç

Yukarıdaki adımları izleyerek, görüntü işleme ve KNN algoritması ile yapılan sınıflandırma sürecinde önemli bilgiler elde ettim. Karşılaşılan sorunlar, kodun gelişimi ve doğruluğun artırılması için çözüm yolları ile birlikte başarılı bir proje ortaya koymayı başardım. Geliştirmeye devam ederek daha iyi sonuçlar almak mümkün olacaktır.

